Reserved Cells

Ex. 1	
Ex. 2	
Ex. 3	
Ex. 4	
Ex. 5	
Ex. 6	
Tot.	

Algorithms and Programming

21 June 2018

Part I: Theory

Register Number _____ Family Name _____ First Name _____

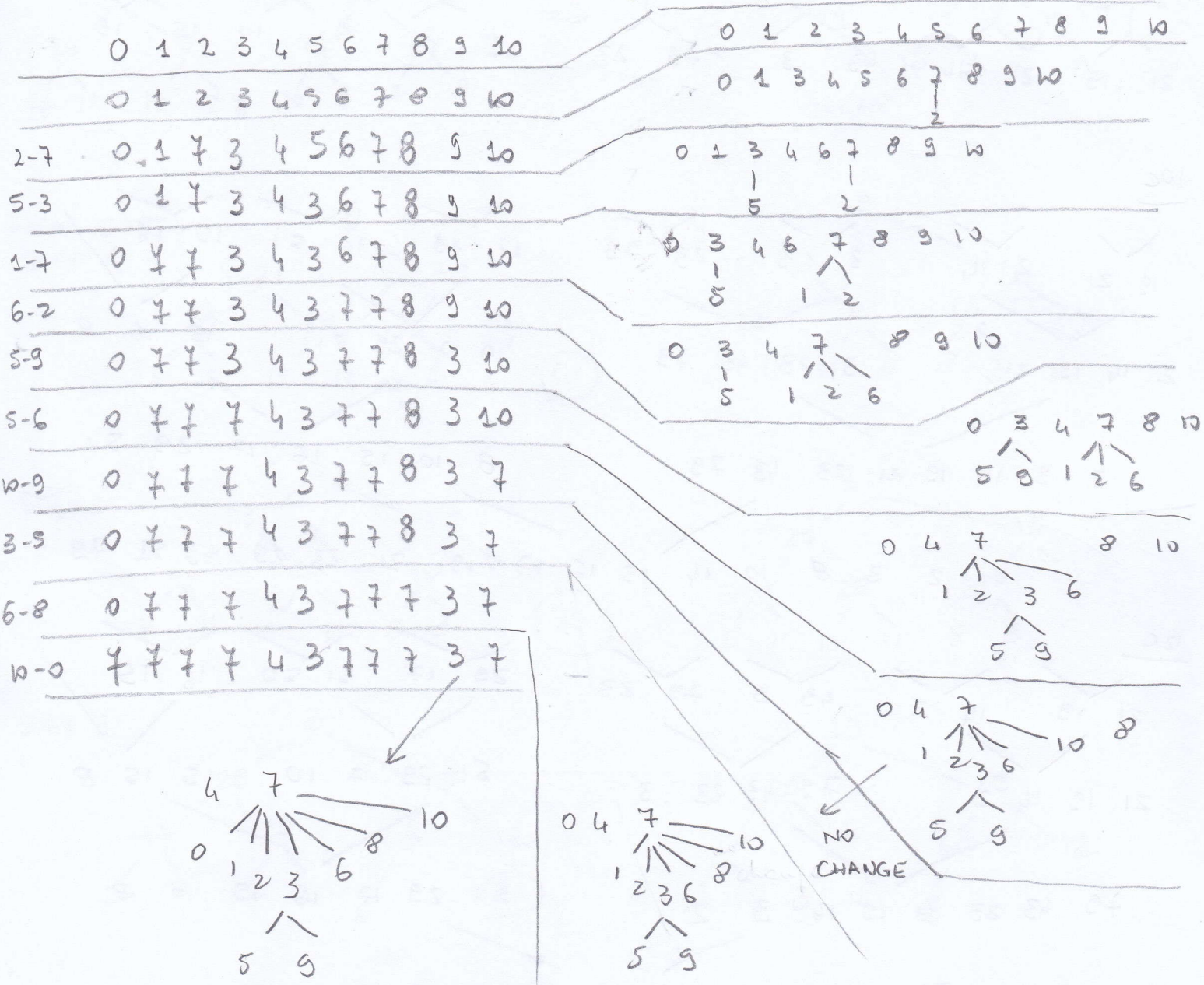
Course: ◯ 10 credit course (01*OGDLP*) ◯ 12 credit course (02*OGDLM*)

No books or notes are allowed. Solve exercises directly within the reserved space. Additional sheets are accepted only when strictly necessary. Examination time: 50 **minutes.**

1. (2.0 **points**)
Given the following sequence of pairs, where the relation i-j means that node i is adjacent to node j:

2-7 5-3 1-7 6-2 5-9 5-6 10-9 3-5 6-8 10-0

apply an on-line connectivity algorithm with weighted quick-union, showing at each step the content of the array and the forest of trees at the final step. Node names are integers in the range from 0 to 10.



2. (1.0 **points**)

10 **credit course** (01*OGDLP*)

Sort in ascending order with merge sort the following array of integers:

21 19 2 14 43 3 79 23 29 17 51 10 15 16 8

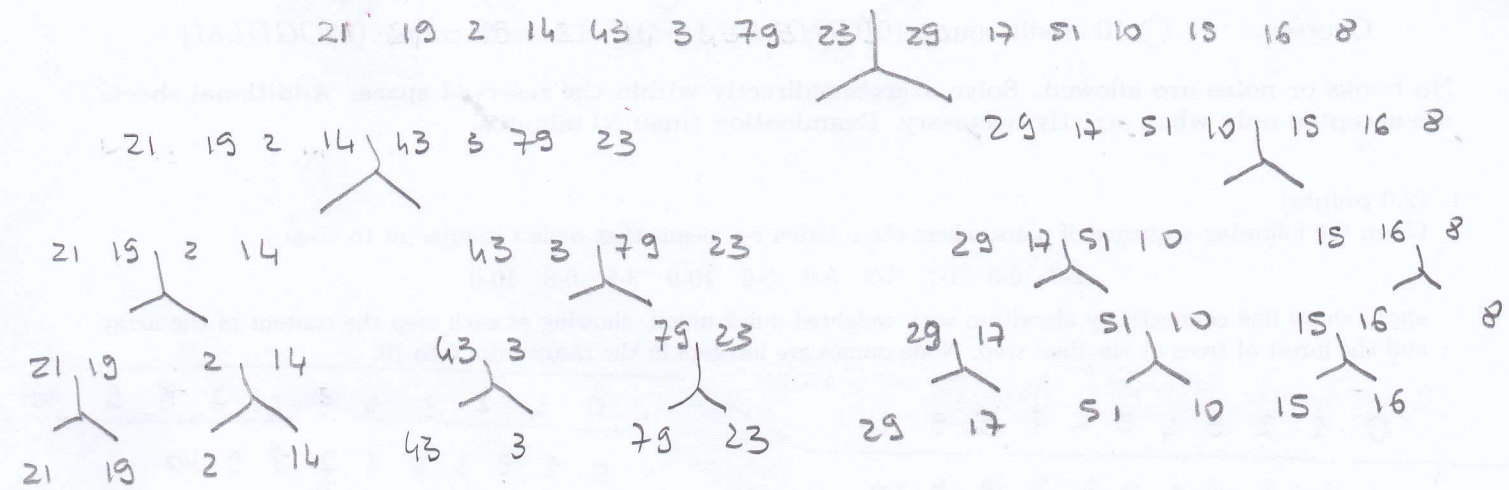
Show relevant intermediate steps.

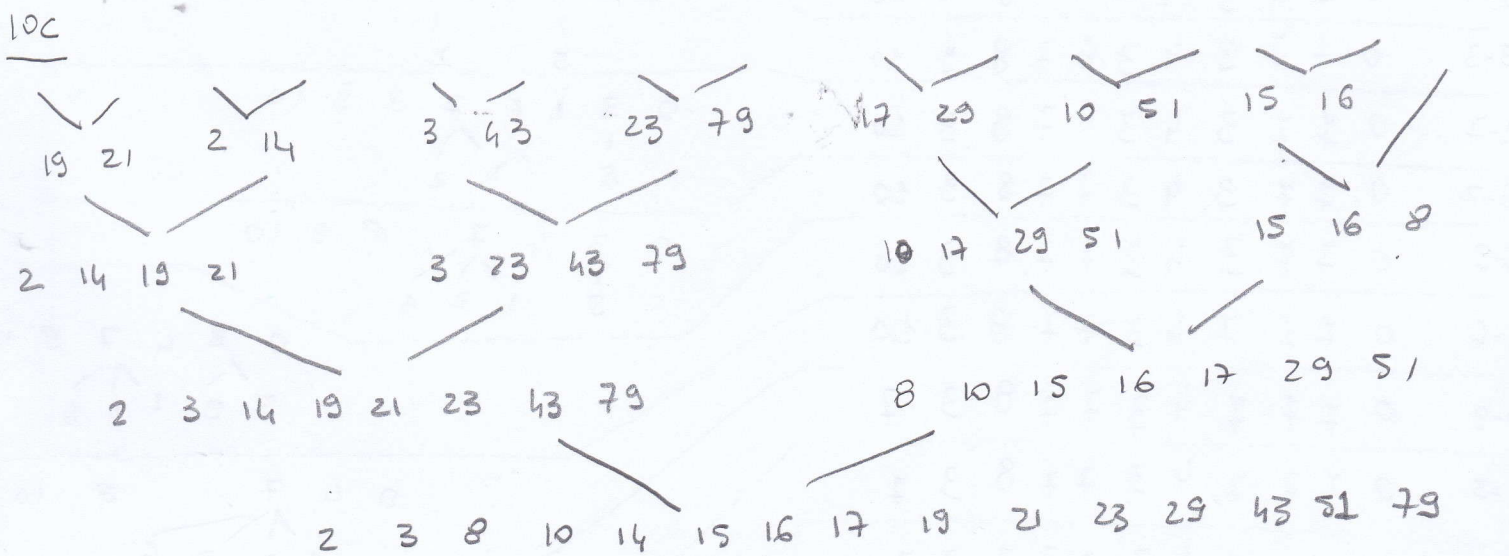
12 **credit course** (02*OGDLM*)

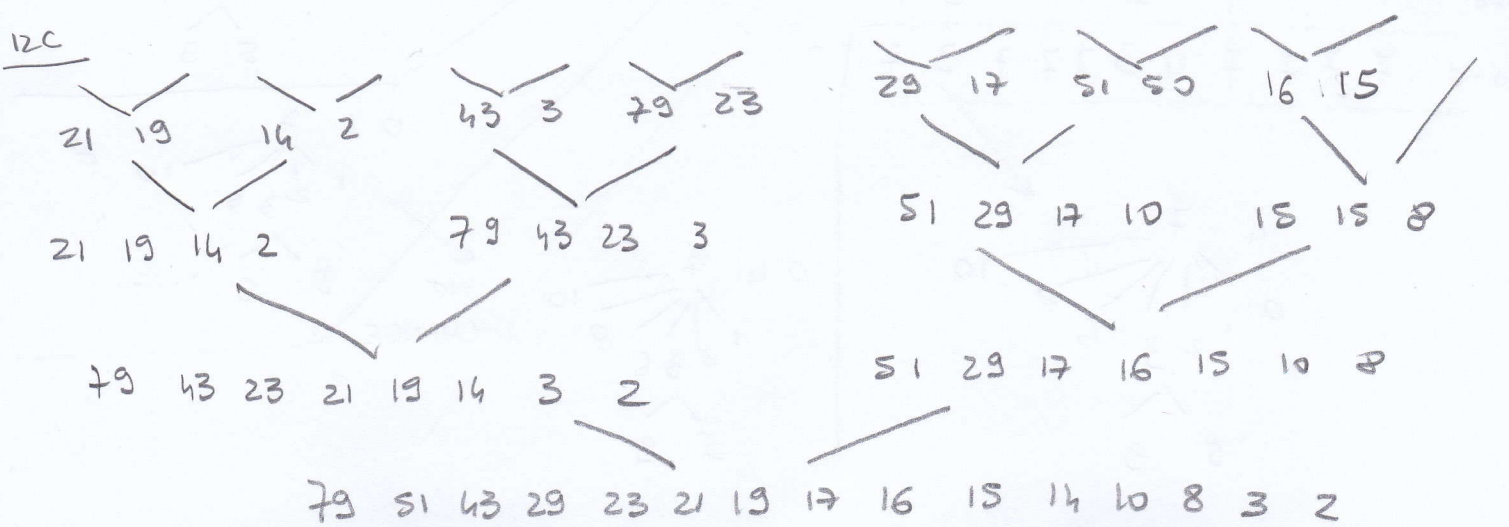
Sort in descending order with merge sort the following array of integers:

21 19 2 14 43 3 79 23 29 17 51 10 15 16 8

Show relevant intermediate steps.





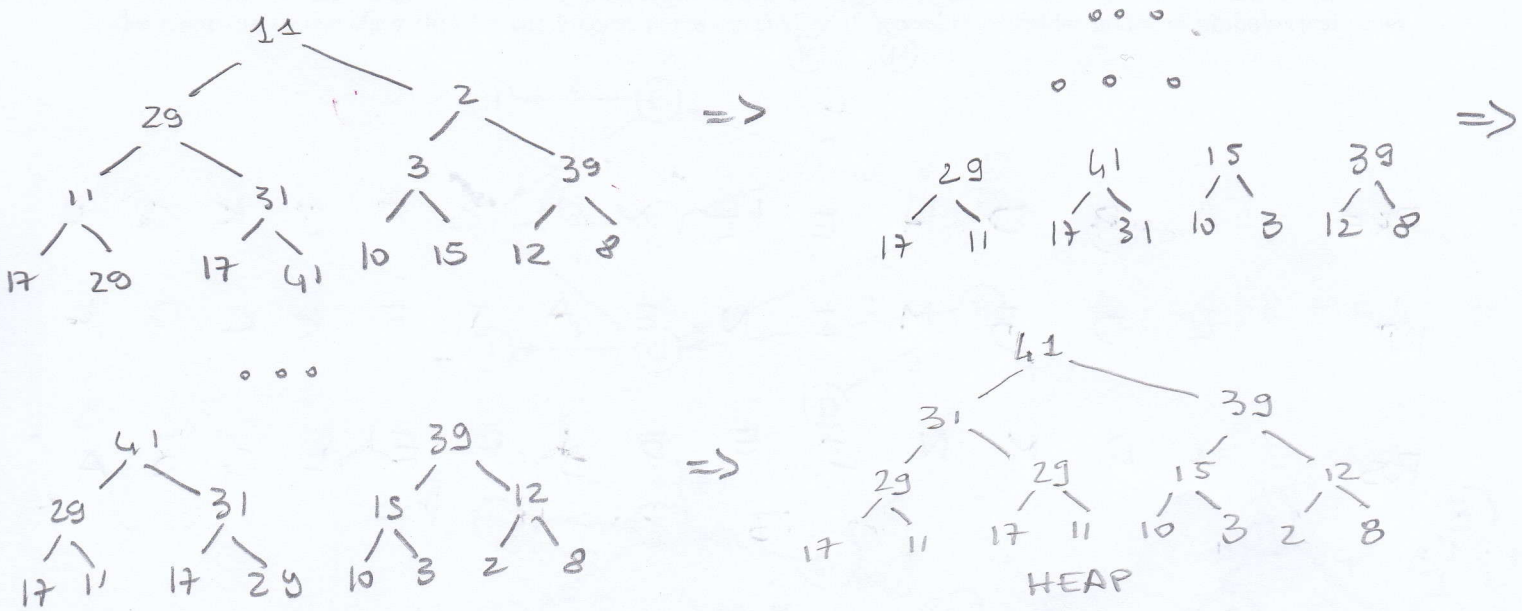


3. (2.5 **point**)
Given the following sequence of integers stored in an array:

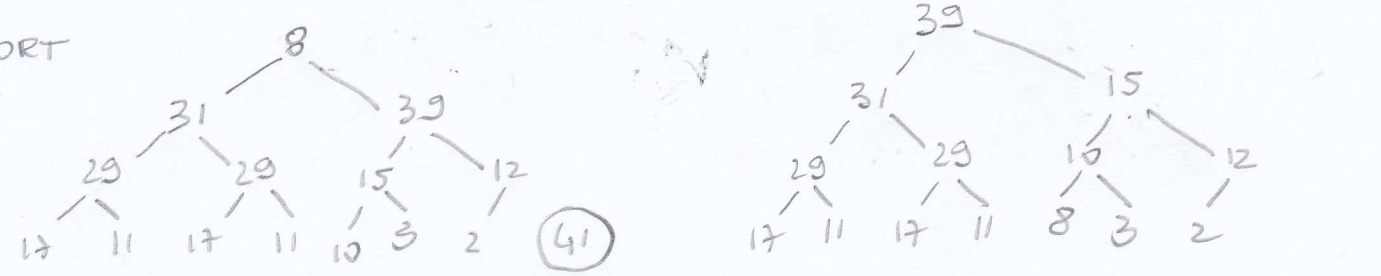
11 29 2 11 31 3 39 17 29 17 41 10 15 12 8

turn it into a heap, assuming to use an array as underlying data structure. Draw each step of the heap-building process, as well as the final result. Assume that, at the end, the largest value is stored at the heap's root. Execute the first three steps of the heap-sort algorithm on the heap built at the previous step.

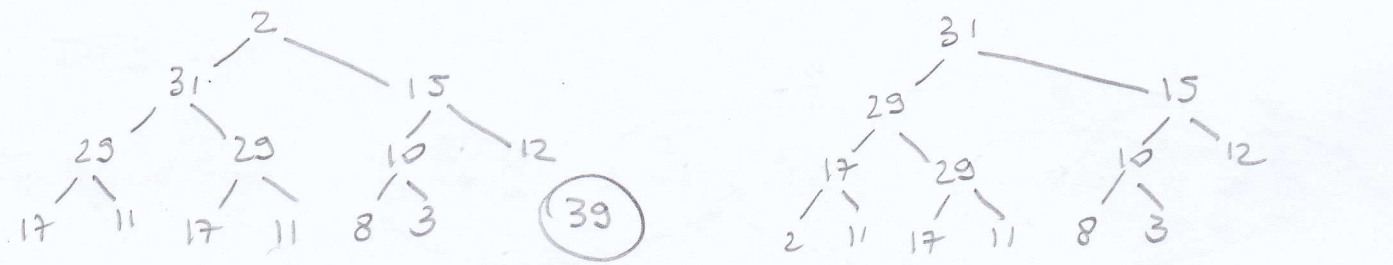
Assume that the sequence is already stored in the array and that it represents an intermediate configuration on which the heap property doesn't necessarily hold.



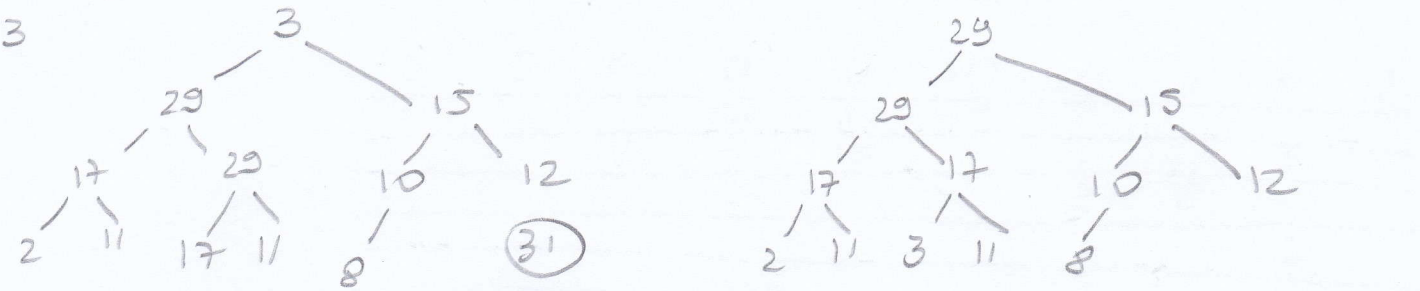
HEAP SORT
STEP 1



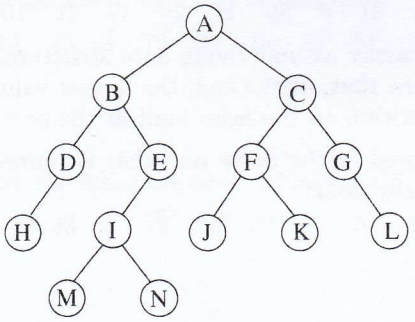
STEP 2



STEP 3



4. (1.5 **points**)
Visit the following binary tree in pre-order, in-order and post-order.



PRE A B D H E I M N C F J K G L

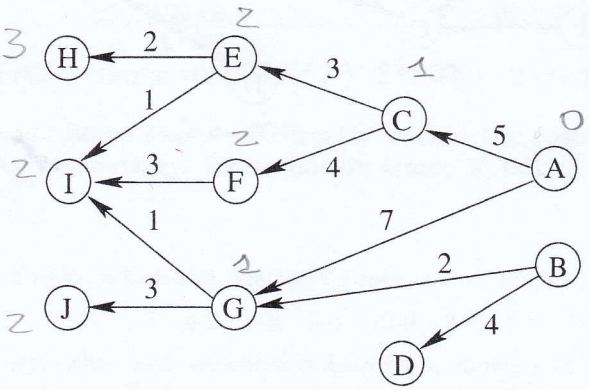
IN H D B M I N E A J F K C G L

POST H D M N I E B J K F L G C A

5. (2.5 **points**)
10 **credit course** (01*OGDLP*)
Consider the following weighted and directed graph. Visit it in breadth-first starting at node A. Label nodes with discovery times.

Redraw it, and visit it in depth-first starting at node A. Label nodes with discovery and end-processing times in the format $time_1/time_2$. Redraw it labeling each edge as T (tree), B (back), F (forward), C (cross). If necessary, consider nodes in alphabetical order.

12 **credit course** (02*OGDLM*)
Consider the following weighted DAG. Starting from A, find all longest paths connecting node A with all the other nodes resorting to the algorithm for the longest paths on DAGs. If necessary, consider nodes in alphabetical order.

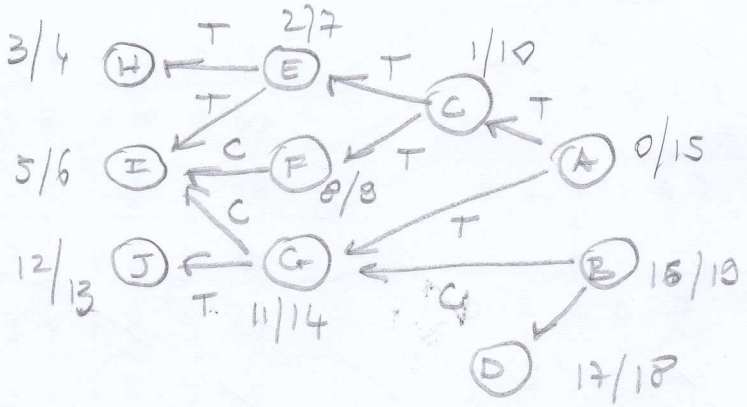


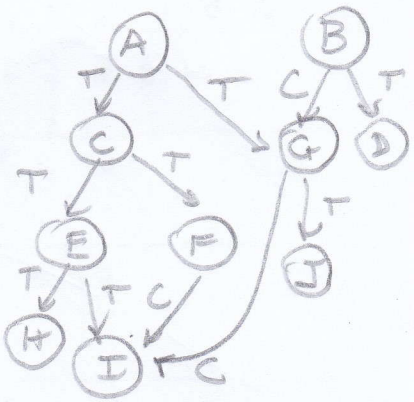
10C

BFS

A | CG | E F I J | H

DFS

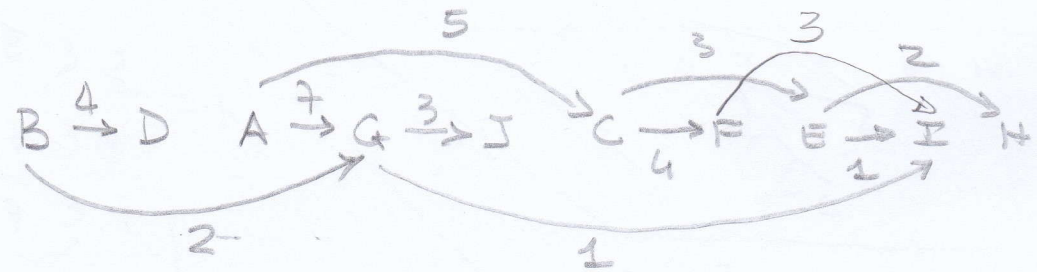


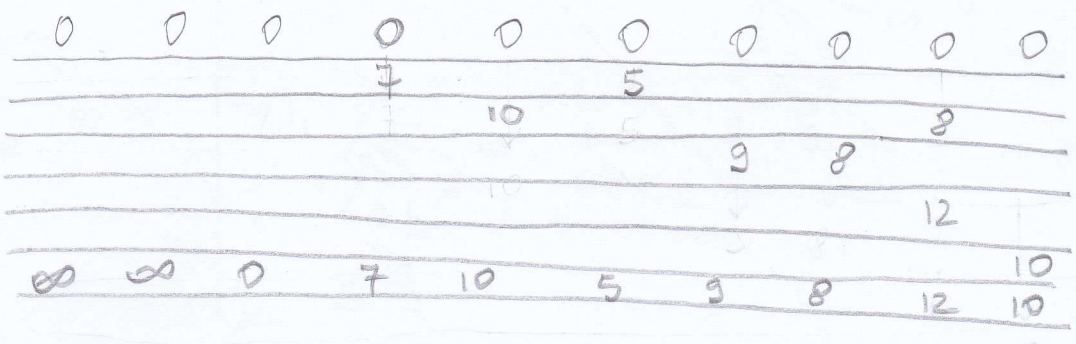


12C

DFS

Topological Sort





6. (2.5 **points**)
On the following directed and weighted graph, find all shortest paths connecting node A with all the other nodes resorting to Dijkstra's algorithm. If necessary, consider nodes in alphabetical order.